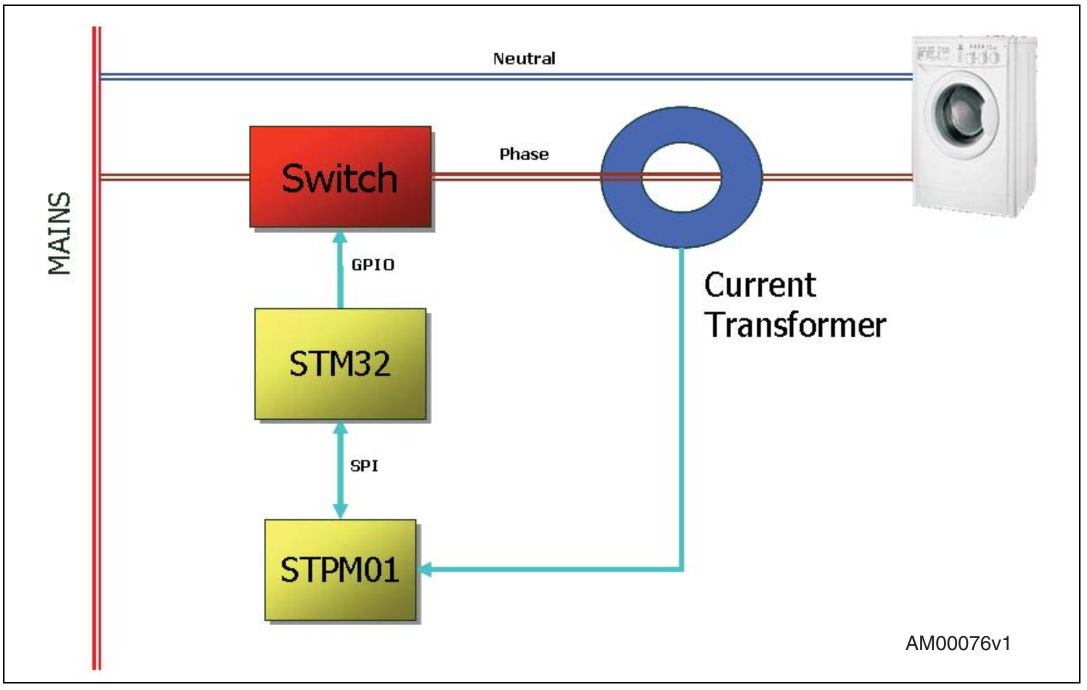
AN2799
Application note

Measuring mains power consumption with the STM32x and STPM01

Introduction

This document describes a software and hardware solution concerning the STM32x microcontroller and the STPM01 power meter for measuring mains power consumption and also provides hardware and firmware guidelines to interface the STPM01 with the STM32x microcontroller. *Figure 1* shows the block diagram of the solution. The system described in this document is the first “brick” to build a complex system for distributed load management.

Figure 1. Block diagram



Contents

1 File organization

The following table presents the firmware modules.

Table 1. Firmware modules

File	Description
PowerMeterHal.h PowerMeterHalTypes.h	Definitions of power meter hardware abstraction layer types and function prototypes.
PowerMeterObj.h PowerMeterTypes.h	Definitions of power meter abstract objects and types.
PowerMeterHal.c PowerMeterObj.c	Power meter hardware abstraction layer and abstract object implementation
Meterlayer.h	Definitions of application interface types and function prototypes
Meter.c MeterHal.c	Application interface and hardware abstraction layer implementation
ExampleApp.c	Library usage example

2 Hardware

The solution is based on the STM32F103xx in the LQFP64 package with 128 kB of Flash and 20 kB of SRAM for demonstration purposes, but it can be easily ported on the smaller STM32F101xx microcontroller family based on the same ARM Cortex™-M3 CPU as well. The device used to measure the phase AC current is the STPM01 with a current transformer.

2.1 STM32F103xx microcontroller

The following list is a brief description of the features of the STM32F103xx microcontroller. Please refer to the STM32F103xx datasheet for a more detailed description of the device.

- Core: ARM 32-bit Cortex™-M3 CPU
  - 72 MHz, 90 DMIPS with 1.25 DMIPS/MHz
  - Single-cycle multiplication and hardware division
- Memory
  - 32-to-128 Kbytes of Flash memory
  - 6-to-20 Kbytes of SRAM
- Clock, reset and supply management
  - 2.0 to 3.6 V application supply and I/Os
  - POR, PDR, and programmable voltage detector (PVD)
  - 4-to-16 MHz quartz oscillator
  - Internal 8 MHz factory-trimmed RC
  - Internal 40 kHz RC
  - PLL for CPU clock
  - 32 kHz oscillator for RTC with calibration
- Low power
  - Sleep, stop and standby modes
  - VBAT supply for RTC and backup registers
- 2 x 12-bit, 1 µs A/D converters (16-channel)
  - Conversion range: 0 to 3.6 V
  - Dual-sample and hold capability
  - Temperature sensor
- DMA
  - 7-channel DMA controller
  - Peripherals supported: timers, ADC, SPIs, I^2Cs and USARTs
- Debug mode
  - Serial wire debug (SWD) & JTAG interfaces
- Up to 80 fast I/O ports
  - 26/36/51/80 I/Os, all mappable on 16 external interrupt vectors, all 5 V-tolerant except for analog inputs
- Up to 7 timers
  - Up to three 16-bit timers, each with up to 4 IC/OC/PWM or pulse counter
  - 16-bit, 6-channel advanced control timer: up to 6 channels for PWM output dead time generation and emergency stop
  - 2 watchdog timers (independent and window)
  - SysTick timer: a 24-bit down counter
- Up to 9 communication interfaces
  - Up to 2 I^2C interfaces (SMBus/PMBus)

- Up to 3 USARTs (ISO 7816 interface, LIN, IrDA capability, modem control)
- Up to 2 SPIs (18 Mbit/s)
- CAN interface (2.0B Active)
- USB 2.0 full speed interface
- Packages are ECOPACK® (RoHS compliant)

2.1.1 STPM01

The STPM01 is a programmable single-phase energy metering IC which is designed for effective measurement of active, reactive and apparent energy in a power line system using a Rogowski coil, a current transformer and shunt sensors. The solution presented in this document is based on a current transformer sensor. The following list is a brief description of the main features of the STPM01. For more details, please refer to the device datasheet.

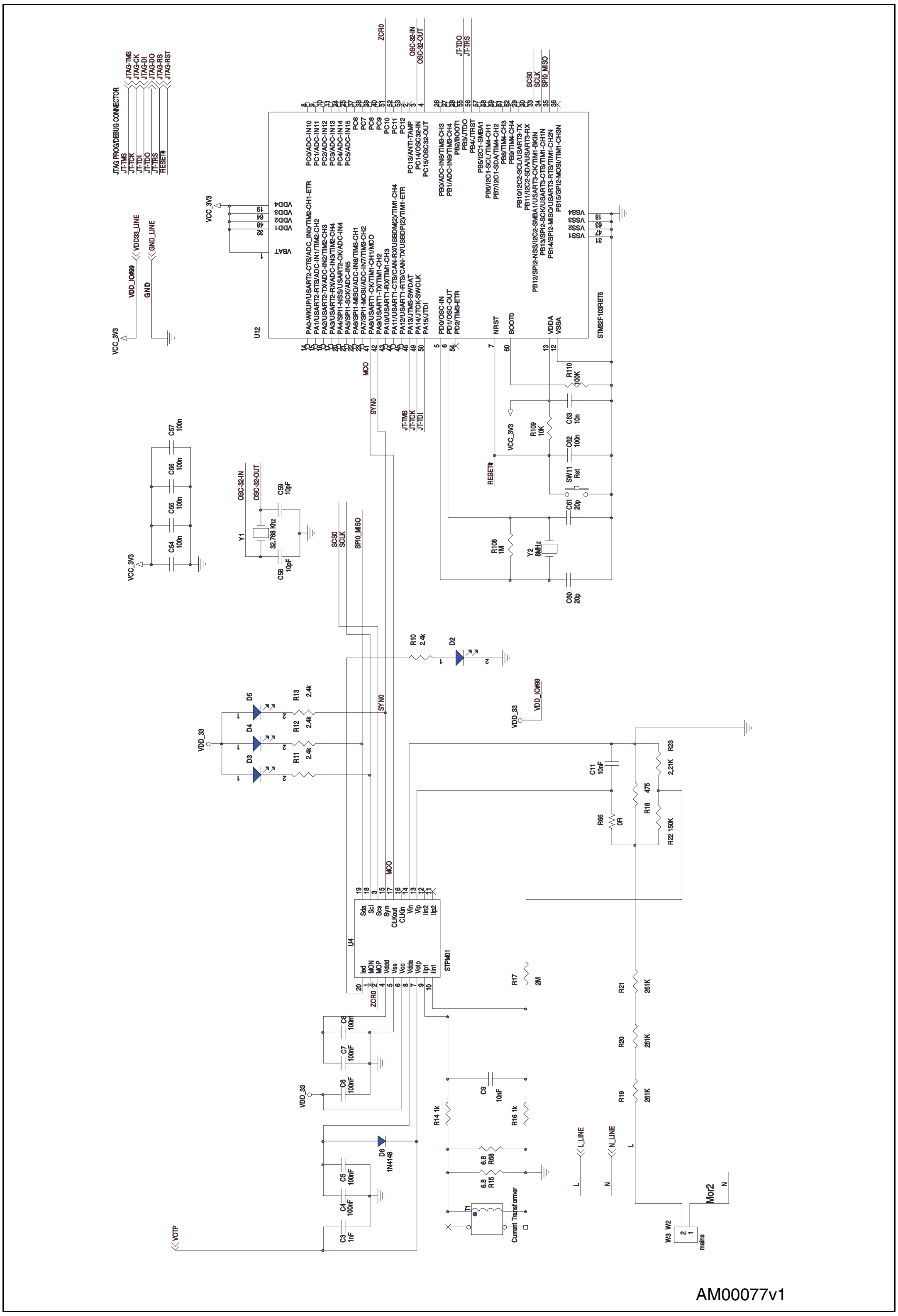
- Active, reactive, apparent energies and RMS values
- Ripple-free active energy pulsed output
- Live and neutral monitoring for tamper detection
- Easy and fast digital calibration in only one point over the whole current range
- OTP for calibration and configuration
- Integrated linear V_{REGS} for digital and analog supply
- Selectable RC or crystal oscillator
- Supports 50-60 Hz - IEC62052-11, IEC62053-2X specification
- Less than 0.1% error
- Precision voltage reference: 1.23 V and 30 ppm/°C max

2.2 Hardware description

The hardware used in managing the STPM01 by the STM32x microcontroller is described in this section. *Figure 2* shows a reference schematic to interface the STM32x microcontroller with the STPM01. The STPM01 bidirectional data line is connected to the SPI-MISO of the STM32. This is because the speeds for the data read and data write operations of the STPM01 SPI interface are much too different: 32 MHz for the read operation and 100 kHz for the write operation. In fact during the write operation, the serial communication is emulated by software to have very slow communication on the STPM01 data line and during the read operation, the STM32 SPI peripheral is used to reach the maximum speed possible. Since the maximum speed of the STM32 SPI is 18 MHz, it doesn't matter which SPI peripheral is used to have the maximum read bit rate. In this application the SPI2 is used. The STPM01 clock is provided by the PA8 pin configured as the MCO alternate function. The AC current is provided to the STPM01 by a current transformer connected to the current channel 1. For more details about analog parts related to the STPM01, please refer to the STPM01 datasheet.

Note: The solution has been validated using the "SmartPlug" board. For further details please refer to the "SmartPlug hardware user manual" .

Figure 2. Schematic diagram



3 Functional description

The STPM01 host interface is a single bidirectional data line SPI with a 32 MHz maximum read speed and 100 kHz maximum write speed. Due to these requirements the STM32 microcontroller is used in the following way:

- In the reading phase, the MISO and SCLK pins are configured in alternate functions and used by the SPI2 peripheral. The maximum reading speed is 18 MHz by an APB speed equal to 36 MHz, configuring the SPI baud rate with APB/2.
- In the writing phase the MISO and SCLK pins are configured as GPIOs and the SPI operation is emulated by firmware controlling the speed operations to not exceed the 100 kHz writing speed limit. The MCO pin is configured to provide the HSE clock.

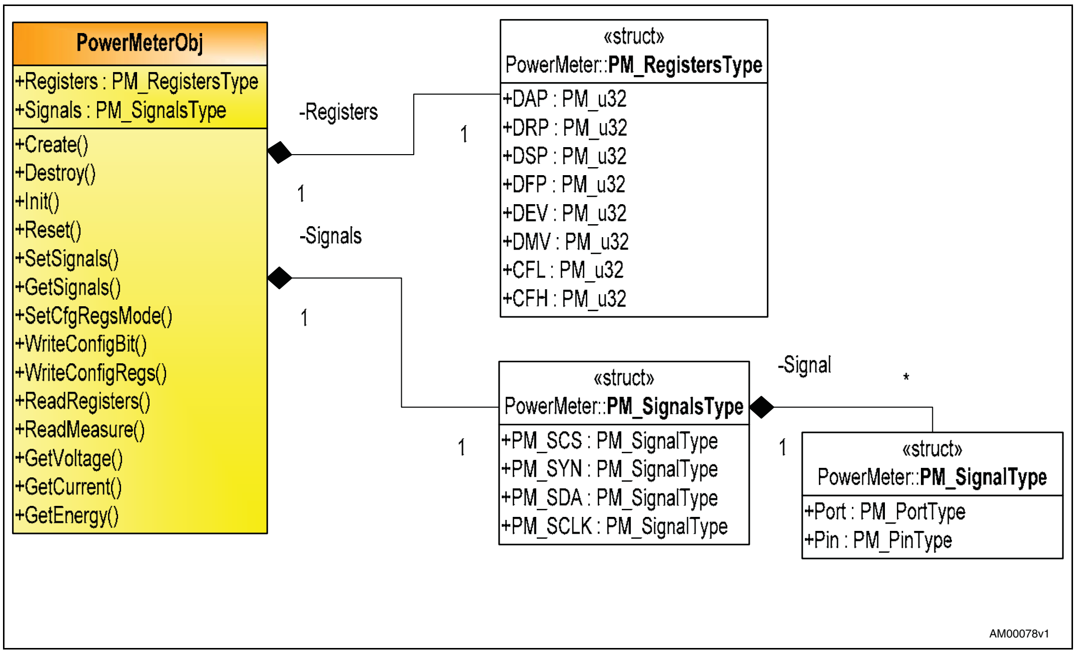
4 Firmware description

The firmware for the management of the STPM01 energy meter has been developed using an object oriented programming (OOP) approach even if standard ANSI C programming language has been used.

4.1 Power meter object description

The power meter object is defined and implemented in the PowerMeterObj.c, PowerMeterObj.h and PowerMeterTypes.h files. The power meter is represented by a structure containing the properties and the method of the Power Meter Object as illustrated in *Figure 3*.

Figure 3. Power meter object



POWER_METER_OBJ

```
#define POWER_METER_OBJ
    PM_u32            m_Voltage;
    PM_u32            m_Current;
    PM_u32            m_Energy ;
    PM_RegistersType Registers;
    PM_SignalsType   Signals;
    PM_ErrStatus     (* Create)(PowerMeterType*);
    PM_ErrStatus     (* Destroy)(PowerMeterType*);
    PM_ErrStatus    (* Init)(PowerMeterType*);
    PM_ErrStatus    (* Reset)(PowerMeterType*);
    PM_ErrStatus   (*SetSignals)(PowerMeterType*,PM_SignalsType* pSignals);
    PM_SignalsType*  (*GetSignals)(PowerMeterType*);
    PM_ErrStatus(*SetCfgRegsMode)(PowerMeterType*,PM_CfgRegsModeType CfgRegsMode);
    PM_ErrStatus(*WriteConfigBit)(PowerMeterType*, PM_u8 uBitAddress, PM_u8
uBitValue);
    PM_ErrStatus(*WriteConfigRegs)(PowerMeterType*,PM_u32 uCFL_Value, PM_u32
uCFH_Value);
    PM_ErrStatus     (*ReadRegisters)(PowerMeterType*, PM_u8 RegsToReadNum);
    PM_ErrStatus     (*ReadMeasure)      (PowerMeterType*, PM_u8 MeasureType);
    PM_u32           (*GetVoltage)(PowerMeterType*);
    PM_u32           (*GetCurrent)(PowerMeterType*);
    PM_u32           (*GetEnergy)(PowerMeterType*);
    PM_ErrStatus     (*OnSpiRxIrq)(PowerMeterType*);


typedef struct PowerMeter  PowerMeterType;  /* Forward declaration for circular
typedefs */
struct PowerMeter {
    POWER_METER_OBJ
};
```

4.1.1 Attributes

The object attributes are described as follows:

- m_Voltage: “private” variable containing the last read voltage in mV
- m_Current: “private” variable containing the last read current in mA
- m_Energy: “private” variable containing the last read energy in mW
- Registers: “public” structured variables containing the STPM01 registers:
  - DAP: type0 active energy
  - DRP: reactive energy
  - DSP: apparent energy
  - DFP: type1 energy
  - DEV: RMS values of measured voltage and current
  - DMV: instantaneous values of measured voltage and current
  - CFL: lower part of the configuration register
  - CFH: higher part of the configuration register
- Signals: “public” structured variables containing the signals descriptor used to interface STM32 with STPM01:
  - PM_SCS: this signal enables the SPI operation when low
  - PM_SYN: When SCS is low, this signal selects the read or write operation. When both SCS and SYN signals are high, the results of the input or output data are transferred to the transmission latches.
  - PM_SDA: SPI MISO signal. This signal is used as the STPM01 data signal. The direction of this signal depends on the SYN signal.
  - PM_SCLK: SPI clock signal

Each signal descriptor contains the STM32 port and pin number. For further details about SPM01 registers and the microcontroller interface, please refer to the STPM01 datasheet.

4.1.2 Methods

The object methods are implemented by means of pointers to functions which are described as follows:

- Create: this function initializes all power meter object elements (properties and methods).
- Destroy: not used. Implemented to have a general object implementation.
- Init: this function initializes the power meter object properties and the power meter signals putting them in the idle state.
- Reset: this function executes the reset sequence to perform the “remote reset” operation of the STPM01 (*Figure 4*).
- SetSignals: this function initializes the “signals” attribute of the power meter object and configures the GPIOs related to the signals.
- GetSignals: this function retrieves the “signals” attribute of the power meter object.
- SetCfgRegMode: this function sets the STPM01 configuration register as shadow latches or OTP. In current AN firmware the STPM01 is configured to use the shadow latches for the configuration for testing reasons.

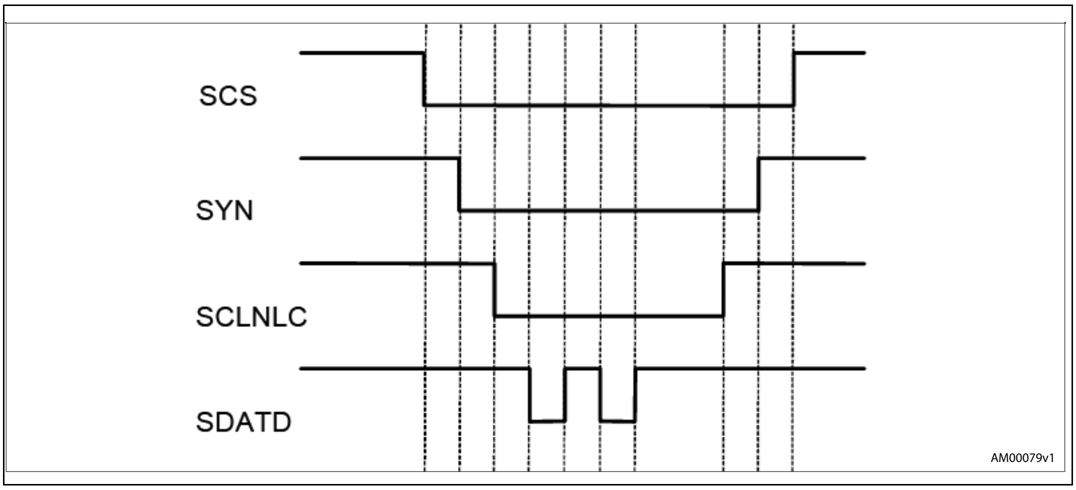
- WriteConfigBit: this function writes a configuration bit of the STPM01 configuration register formatting an 8-bit command word as the following:

1 bit data	6 bits configuration bits	1 bit don't care

The configuration word is sent to the STPM01 by the SPI interface.

- WriteConfigRegs: this function writes the entire CFL and CFH configuration registers of the STPM01.
- ReadRegisters: this function reads the registers from the STPM01 and assigns the read values to the power meter object registers attribute. The number of read registers depends on the RegToReadNum passed parameter.
- ReadMeasure: this function reads the RMS voltage and current registers from the STPM01, calculates the real RMS values by multiplying the values of each register by the calibration factors and assigns these values to the m_Voltage and m_Current attributes of the power meter object.
- GetVoltage: this function retrieves the last read voltage value in mV.
- GetCurrent: this function retrieves the last read current value in mA.
- GetEnergy: this function retrieves the last read energy value in mW.
- OnSpiRxIrq: SPI interrupt event. Not used in this implementation. The SPI interrupt is managed externally by the power meter object implementation.

Figure 4. Reset sequence



4.1.3 Construction

The power meter object is constructed by the NewPowerMeterObj function implemented externally by the object itself. This function allocates the memory for the object, assigns the create method and calls it to assign the rest of the object properties.

4.1.4 Deletion

The power meter object is deleted by the DeletePowerMeterObj function implemented externally by the object itself. This function first calls the deletion method to free dynamically allocated variables during object creation (in this case does nothing) then frees the allocated memory for the power meter object.

4.2 Hardware abstraction layer

The HAL is implemented using the following files:

- PowerMeterHal.c
- PowerMeterHal.h
- PowerMeterHalTypes.h

This module is useful to abstract the power meter object implementation from the STM32 standard library.

4.2.1 HAL functions

The hardware abstraction layer functions are described as follows:

- PM_ConfigSignal: this function configures a pin according to the passed signal descriptor and configuration type.
- PM_ReceiveRegOnSpi: this function receives the STPM01 internal register from the SPI interface.
- PM_OnSpiRxIrq: This function implements the interrupt service routine of the SPI interface. It is called when the FIFO of the SPI peripherals contains 1 byte or more and assigns the received value to a shared register variable.
- SendClkToGetData: this function is used to provide the SPI clock to the STPM01 to receive data.
- ConfigPin: pin configuration function.

4.3 Application interface

The application interface is implemented using the following files:

- Meter.c
- MeterHal.c
- MeterLayer.h

This layer is useful to abstract the application from the specific board used. In fact, the functions implemented in the described modules allow the hardware configuration of the specific board used. The firmware described in this document has been validated using the "SmartPlug" board that includes two STPM01 meters enumerated as PM_METER_TYPE_CONSUMPTION and PM_METER_TYPE_DISPERSION.

4.3.1 Application interface functions

The application interface functions are described as follows:

- PM_HwMetersCommonInit: this function configures the SPI peripheral (including the interrupt controller), the STM32 MCO pin to provide the clock to both STPM01 and a GPIO as input, useful to select two different operational modes:
  - Normal Mode: typical operational mode.
  - Calibration Mode: used to calibrate the STPM01 devices by an external tool. In this mode the STM32 provides only the clock to the STPM01 devices.

These configurations are common to both STPM01 devices.

- M_HwMeterSignalsInit: this function initializes the signals specific for an STPM01. In detail, the STM32 GPIO is used as the SCS for the specific device. The specific STPM01 device is selected by an enumerator as already stated.
- ActivateMeter: this function includes all the operations to be done using the power meter object for the STPM01 initialization.

All the initialization operations have been included in a single initialization procedure: Init, establishing a power meter object and the STPM01 device identifier as parameters.

5 Conclusion

Summarizing, this document explains how to interface the STPM01 energy meter device to the STM32x microcontroller including an easy-to-use firmware library. The firmware library also includes the file ExampleApp.c which shows how to use the library itself. The user application has to perform the standard STM32 configuration and simply call the NewPowerObj for each STPM01 device to allocate the object, call the Init to perform the power meters related initializations and configurations, call the ReadMeasure method to get the data from STPM01 device and call GetVoltage and/or GetCurrent to get the last measurement data.

6 Revision history

Table 2. Document revision history

Date	Revision	Changes
27-Nov-2008	1	Initial release

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2008 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com